Math 540 - Assignment 3

Lewis Blake

April 11, 2019

1 Introduction

Let $P = mN$ be the total number of processing cores in `MPI_COMM_WORLD` induced by `mpiexec`, with N and m respectively being the number of nodes and cores in each node. Let `master` $= 0$ denote the first processing core in `MPI_COMM_WORLD`.

Let $J \geq 2$ and $L = JP$. Let C be a random $L \times L$ weighted covariance matrix (described below) with each processing core generating only J distinct random rows of C. Let

$$C_{\max} = \max\{C_{i,j} : 1 \leq i, j \leq L\}, \quad C_{\min} = \min\{C_{i,j} : 1 \leq i, j \leq L\}.$$

Let $\text{core}_{\max}$ be the number of the processing core that (with minimum index), that first generated the number $C_{\max}$ in `MPI_COMM_WORLD`. Let the location of $C_{\max}$ in C be (i_{max}, j_{max}). Let $\text{core}_{\min}$ be the number of the processing core that (with minimum index), that first generated the number $C_{\min}$ in `MPI_COMM_WORLD`. Let the location of $C_{\min}$ in C be (i_{min}, j_{min}).

The main tasks of this MPI programming assignment include the `master` to search and find the pairs ($C_{\max}$, $\text{core}_{\max}$) and ($C_{\min}$, $\text{core}_{\min}$) in `MPI_COMM_WORLD` and hence identify the pairs (i_{max}, j_{max}) and (i_{min}, j_{min}).

Your code should be such that it is easier for you / the grader to choose $J = 2$ and run your code with $P = 4$ to validate the code. Only for this special case, your code should be such that the `master` should print

- the matrix C (one row per line and easy to read to validate the vectors below);
- the vectors $(C_{\max}, \text{core}_{\max}), (C_{\min}, \text{ core }_{\min}), (i_{\max}, j_{\max}),$ and $(i_{\min}, j_{\min})$

If $J \neq 2$ or $P \neq 4$ your code should be such that the `master` should print

- the vectors $(C_{\max}, \text{core}_{\max}), (C_{\min}, \text{core}_{\min}), (i \max, j \max),$ and $(i \min, j \min)$
- the MPI walltime required to execute lines between `MPI_Init` and `MPI_Finalize`.

Use suitable format and headings for your print so that the output can be easily read and identified. Submit your output for the following choices of J and P:

(i) $J = 2, P = 4$; (ii) $J = 100, P = 16$; (iii) $J = 50, P = 32$; and (iv)$J = 25, P = 64$.

The main (memory) constraint in your parallel program with MPI to achieve the above tasks is that at any point in executing your code (for any choices of J and P), each processing core should contain at most J consecutive rows of the covariance matrix C. (Only exception in your code is for the $J = 2$, $P = 4$ case so that the `master` can gather and print C.)

2 Procedure and Program Parallelization

The procedure to generate the random weighted covariance matrix is broken up into two steps.

Step 1.
For $k = 1, ..., P$, the k^{th} processing core generates the a J-dimensional random weight vector $\mathbf{r}_k$ so that for $i = 1, ..., J$, the i^{th} entry of $\mathbf{r}_k$ is in the range $[k/i, i * k + 1)$. The `master` gathers these random weight numbers via `MPI_Gather` and computes

$$S = \sum_{k=1}^{P} \sum_{j=1}^{J} r_{k,j}.$$

The `master` then computes an L-dimensional random weight vector $\mathbf{w}$ with the first J entries in $\mathbf{w}$ being $r_{1,i}/S, \quad i = 1, \cdots, J$; the $J+1, \cdots, 2J$ entries being $r_{2,i}/S, \quad i = 1, \cdots, J$, etc. so that the last J entries of $\mathbf{w}$ are given by $r_{P,i}/S, i = 1, \cdots, J$. If $\sum_{j=1}^{L} w_i \neq 1$ (within tolerance), the program calls `MPI_Abort` and the program stops with a message that this requirement was not satisfied.

The `master` then broadcasts the normalized weight vector $\mathbf{w}$ to all processing cores in `MPI_COMM_WORLD` using `MPI_Bcast`. A call to `MPI_Barrier` is called prior to `MPI_Bcast` to ensure `MPI_Bcast` is called on all cores simultaneously.

Step 2.
Let X be an $L \times L$ random sample matrix with (i, j)-th entry, $X_{i,j}$ being a random number in the range $[-i * j, i/j)$, for $i, j = 1, \cdots, L$. The L sample means of X are given by

$$\overline{x}_i = \frac{1}{L} \sum_{k=1}^{L} X_{i,k}, \quad i = 1, \cdots, L.$$

Using **Step 1** and X, the (i, j)-th of the weighted $L \times L$ covariance matrix C is defined as

$$C_{i,j} = \frac{\sum_{k=1}^{L} w_k \left(X_{i,k} - \overline{x}_i\right) \left(X_{j,k} - \overline{x}_j\right)}{1 - \sum_{n=1}^{L} w_n^2}, \quad i, j = 1, \cdots, L.$$

Each processing core allocates space for two $J \times L$ matrices for creating J rows of X and C such that the first processing core generates the first J rows of X; the second processing core generates rows $J + 1, \cdots, 2J$ of X; etc. so that the last J rows of X are generated by the last processing core in `MPI_COMM_WORLD`.

Each processing core then computed J sample means corresponding to the J rows of X that it posses in its local memory. Then each processing core creates J rows of X. Each cores posses the rows needed to create the block-diagonal structure of C, and so this portion of creating C is completed first. Then, to create the off-block-diagonal elements of C, communication between the cores is required. To fill in every sub-block of C, every core except the `master` sends it's rows of X to all cores with lesser rank. This is accomplished by having each core loop through the cores with lesser rank, and sending them its rows of X with a call to `MPI_Send`.

Each core then declares a buffer to receive the rows. Each core then loops through the cores it is expecting to received rows from, receives them by means of `MPI_Recv`, and performs the matrix multiplications needed to create a specific sub-block of C, identifying its corresponding location in C. Finally each core weights its entries of C with the scalar multiple given in $C_{i,j}$ previously.

Once C has been computed (distributed across cores), each core then searches for it's maximal and minal entries, as well as their locations. The `master` then gathers this data by means of `MPI_Gather` and searches through each core to find $C_{\max}$, $\text{core}_{\max}$), ($C_{\min}$, $\text{core}_{\min}$), (i_{max}, j_{max}), and (i_{min}, j_{min}). Results of this search are printed to the console screen. In the special case that $J = 2$ and $P = 4$, all the rows of C are gathered onto the `master` with a `MPI_Gather` call and the C matrix is printed to the console screen.

3 Submitted Results

Following are the submitted results (outputs) for various choices of J and P. Note: $(i_{\max}, j_{\max})$ and $(i_{\min}, j_{\min})$ are given assuming array indices start at zero. Additionally, only one set of coordinates are given. Since C is a symmetric covariance matrix, the same maximums and minimums are obtained in the locations of C given by swapping the row and column coordinates of the extrema presented.

(i) $J = 2, P = 4$:

$(C_{\max}, \text{core}_{\max})$ = (1.4931e+03, 1)
$(C_{\min}, \text{core}_{\min})$ = (-4.3734e+00, 3)
$(i_{max}, j_{max}) = (3, 5)$
$(i_{min}, j_{min}) = (7, 6)$

Note: decimal places in the output below are been truncated for sake of presentation.
The C matrix is:

$$\begin{bmatrix}
5.4837e+00 & 9.6846e+00 & 1.6827e+02 & 1.8176e+02 & 2.7156e+02 & 6.1564e+02 & 6.2628e+01 & 2.2553e+02 \\
9.6846e+00 & 3.1312e+01 & 2.9486e+02 & 2.0098e+02 & 3.9810e+02 & 1.0278e+03 & 8.2006e+01 & 1.6049e+02 \\
1.6827e+02 & 2.9486e+02 & 2.1930e+01 & 3.9492e+00 & 4.5744e+02 & 1.1187e+03 & 1.0784e+02 & 2.2410e+02 \\
1.8176e+02 & 2.0098e+02 & 3.9492e+00 & 5.0243e+01 & 7.3311e+02 & 1.4931e+03 & 1.2436e+02 & 5.2165e+02 \\
2.7156e+02 & 3.9810e+02 & 4.5744e+02 & 7.3311e+02 & 5.4590e+01 & 1.0618e+02 & 1.2658e+02 & 7.5403e+02 \\
6.1564e+02 & 1.0278e+03 & 1.1187e+03 & 1.4931e+03 & 1.0618e+02 & 2.4243e+02 & 3.0480e+02 & 1.4486e+03 \\
6.2628e+01 & 8.2006e+01 & 1.0784e+02 & 1.2436e+02 & 1.2658e+02 & 3.0480e+02 & 5.8864e+00 & -4.3734e+00 \\
2.2553e+02 & 1.6049e+02 & 2.2410e+02 & 5.2165e+02 & 7.5403e+02 & 1.4486e+03 & -4.3734e+00 & 1.0079e+02
\end{bmatrix}$$

(ii) $J = 100, P = 16$:

$(C_{\max}, \text{core}_{\max})$ = (8.4767e+14, 14)
$(C_{\min}, \text{core}_{\min})$ = (1.0622e+05, 0)
$(i_{max}, j_{max}) = (1498, 1598)$
$(i_{min}, j_{min}) = (1, 0)$

MPI walltime: 1.6379e-01

(iii) $J = 50, P = 32$:

$(C_{\max}, \text{core}_{\max})$ = (8.3065e+14, 30)
$(C_{\min}, \text{core}_{\min})$ = (1.1557e+05, 0)
$(i_{max}, j_{max}) = (1549, 1597)$
$(i_{min}, j_{min}) = (1, 0)$
MPI walltime: 2.5165e-01

(iv) $J = 25, P = 64$:

$(C_{\max}, \text{core}_{\max})$ = (8.7659e+14, 62)
$(C_{\min}, \text{core}_{\min})$ = (9.9643e+04, 0)
$(i_{max}, j_{max}) = (1574, 1599)$
$(i_{min}, j_{min}) = (1, 0)$
MPI walltime: 3.4050e-01